

0.1 Tkinter: première approche

Tkinter (de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques (Graphical User Interface ou GUI) basée sur la boîte à outil **Tk**. Vous trouverez une documentation complète en suivant ce lien: <http://tkinter.fdex.eu/doc/sa.html> et des exemples bien posés en suivant celui-ci: <https://python.doctor/page-tkinter-interface-graphique-python-tutoriel>

0.1.1 Programme minimal pour afficher une fenêtre

```
1 #coding: utf-8
2 from tkinter import *
3 fenetre = Tk()
4 ...
5 ...
6 ...
7
8 fenetre.mainloop()
```

L'instruction `fenetre.mainloop()` à écrire en fin de code place la fenêtre créée en attente d'un évènement. Les lignes 4, 5, 6 sont à remplacer par votre code.

0.1.2 Les widget

Il s'agira d'ajouter dans la fenêtre des composants graphiques appelés *widget* ou *textitwindow* gadget. Ce sont des objets comme des boutons, des étiquettes, etc.

```
1 #coding: utf-8
2 from tkinter import *
3 fenetre = Tk()
4 zone_dessin = Canvas(fenetre, width=800, height=500, bg='white', bd=8,
5                       relief="flat", background="white")
6 ...
7 ...
8 ...
9
10 label = Label(fenetre, text="Hello World")
11 label.pack()
12 zone_dessin.pack()
13 bouton_sortie = Button(fenetre, text="Quitter", command=fenetre.quit)
14 bouton_sortie.pack()
15 fenetre.mainloop()
```

La variable `zone_dessin` est un *canevas*: zone de la fenêtre où on peut dessiner. L'appel de la méthode `pack()` assure l'affichage des différents éléments de cette fenêtre.

Le constructeur de la classe `Button` retourne le nouveau widget `Bouton_sortie`. Ses options sont `text` et `command` que l'on comprend facilement, vous remarquerez que l'appel à la fonction `destroy` n'est pas associée à des parenthèses. Consulter la documentation de chaque classe avec `help()`. Le premier paramètre est le **parent** du widget: il faut en effet indiquer qui contient ce widget: ici l'objet instancié `fenetre`.

0.1.3 Les couleurs

Il est possible d'indiquer une valeur de couleur par son nom en anglais: "white", "black", "red", "yellow", etc. ou par son code hexadécimal: #000000, #00FFFF..., etc. Les couleurs sont donc représentée avec **Tkinter comme en HTML** par 3 nombres hexadécimaux représentant les tons

de rouge, de vert et de bleu de la couleur choisie: c'est le codage RVB en français ou RGB en anglais. Ainsi la syntaxe de codage d'une couleur est la suivante :

```
couleur = "#RRVVBB"
```

Le type de cette variable est donc `str` c'est une chaîne de caractère. Utilisez la palette de Gimp ou l'un des nombreux sites comme https://fr.wikipedia.org/wiki/Couleur_du_Web pour manipuler quelques couleurs avec leurs codes RVB et HTML.

Nous voulons générer des couleurs aléatoires avec cette représentation.

A faire

1. Rappeler le calcul de la valeur décimale de FF_{16} .
2. L'instruction `fen = Tk()` affiche une fenêtre. Expliquer avec le vocabulaire du cours POO ce qu'il se produit au niveau du code.
3. Expliquer quelle est la différence entre `label` et `Label` dans l'instruction ci-dessous:

```
label = Label(fenetre, text="Hello World")
```

4. Couleurs aléatoires

Afin de générer des couleurs aléatoirement, nous voulons choisir un triplet de trois entiers de façon aléatoire. Il nous faut pour cela une fonction prenant en paramètre 3 entiers compris entre 0 et 255 et retournant le code hexadécimal lisible par `tkinter/HTML` pour représenter une couleur correspondant à ce **code RVB**. on nomme cette fonction `rvb2html()`. Voici un exemple de ce qui est attendu:

Exemple

```
>>> rvb2html(100, 250, 255)
'#64faff'
```

- (a) Utilisez un `help()` sur la fonction *built-in* `hex()` afin de lire sa documentation puis effectuez quelques tests de cette fonction.

- (b) Ecrire une fonction `dec2hex()` convertissant un entier compris entre 0 et 255 en sa valeur hexadécimale. Vous pouvez utiliser la fonction `hex()` de python mais attention, cette chaîne de caractère devra comporter *exactement* deux caractères.

Exemple

```
>>> dec2hex(100)
'64'
>>> dec2hex(8)
'08'
```

5. Ecrire un programme affichant des lignes de couleur aléatoire et de position aléatoire.

0.1.4 Un peu de programmation événementielle

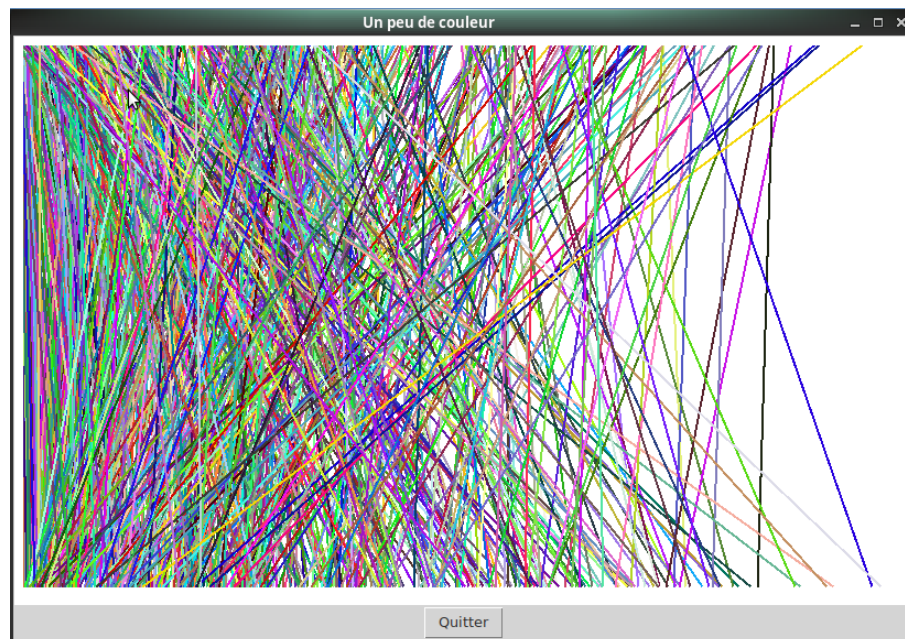
Dans notre fenêtre `fenetre`, l'appui sur une touche est un événement. la méthode `mainloop()` crée une boucle infinie dont on ne sort que lorsqu'un événement précis se produit. On appelle événement toute action de l'utilisateur telles que cliquer ou déplacer la souris, taper au clavier, toucher un écran tactile etc. Pour réagir à ces événements, on utilise des fonctions qui sont à leur écoute, elle vont prendre en paramètre une **fonction de rappel**. Une **fonction de rappel** (*callback* en anglais) est une fonction qui est passée en argument à une autre fonction. Cette-dernière peut alors faire usage de cette fonction de rappel comme de n'importe quelle autre fonction, alors qu'elle ne la connaît pas par avance. `fenetre.bind(événement, fonction)` est un écouteur d'événement. Vous pouvez par exemple utiliser l'événement qui correspond à l'appui sur une touche: "`<KeyPress>`":

```
fenetre.bind("<KeyPress>", appui_touche)
```

A faire: Ecrivez une fonction de rappel `appui_touche(evt)` de telle façon qu' à l'appui sur une touche, une nouvelle image soit générée.

Exemple

```
>>> coul_alea()
```



0.2 Annexes

- Programmation événementielle avec tkinter